A Unified Process for Software and Documentation Development

Michael Priestley
IBM Toronto Lab
1150 Eglinton Ave. E.
Toronto, Ont. M3C 1H7
mpriestley@acm.org

Mary Hunter Utt[1]
SiteScape, Inc.
Ten Clock Tower Place, Suite 100
Maynard, MA 01754
utt@sitescape.com

This paper proposes integration of the documentation development process into the Rational Unified Process (RUP), a formal development process for software applications. Specifically, the paper identifies (in RUP parlance) the workers in the process (such as technical writer, information architect), the artifacts required by and produced by the documentation process (including concept, task, and reference documentation), and the documentation development workflow (the activities of the workers who produce the artifacts).

This paper describes a documentation development process in terms of its integration points with software development processes, and also in terms of its own flow and progression as a separate process.

[1] This paper represents work done while this co-author worked for Rational Software, Inc.

0-7803-6431-7/00/$10.00 © 2000 IEEE

INTRODUCTION

The end-user information, or documentation, delivered with a software product is an integral part of that product. In fact, the product effectively does not exist until its documentation exists. Yet many software development processes overlook the set of activities that constitute user information development. Consequently, user information is often thrown together hurriedly at the end of a project, and thus runs the risk of being incomplete, inaccurate, and unusable.

This is ironic given that the documentation is, along with the user interface, the most user-visible aspect of the software product! In fact, delivering high-quality software documentation is a complex set of activities that parallels the development of the software (or "bits"). It begins with architecture, moves through analysis, design, and implementation, undergoes testing, and finally is delivered to users.

The processes of developing bits and developing docs use many of the same product artifacts, notably use cases (formal descriptions of system behavior in particular usage scenarios), and have many points of intersection, such as reviews. Done well, both begin early in the product planning cycle, or Inception phase of the Rational Unified Process.

The terms *user information* and *documentation* are used more or less synonymously. *Documentation* is a widely understood term, but it often connotes printed manuals. Modern documentation, or *user information*, has a broadening scope. Online help and documentation have been around for a while, and they have been joined by newer technologies such as the World Wide Web and embedded help. The bits and the docs are becoming more tightly integrated; some of the lines between them are getting fuzzier. In this paper, wherever the term documentation is used, it should be understood to include all types of user information, including printed, online, web-based, and embedded.

The Rational Unified Process: A Brief Introduction

The Rational Unified Process (RUP) is a set of recommended best practices for software development. The RUP has four basic concepts: *worker*, *activity*, *artifact*, and *workflow*.

- A *worker*, essentially a role, is a set of behaviors and responsibilities that an individual may carry out. In fact, a worker can represent more than one individual, and one individual can represent more than one worker. Examples: architect, test designer, implementer.
- An *activity* is something a worker does that provides a meaningful result. Examples: developing design guidelines; testing a component.
- An *artifact* is a product of an activity. Examples: design model; component; test procedure.
- A *workflow* is a logical group of activities. Examples: designing; implementing; testing.

These concepts come together as part of a software development process in which workers complete activities to produce artifacts within workflows. For more information on these workflows, see [1] and [2].

We will extend each of the existing development workflows in turn to cover the workers and artifacts required to develop end-user documentation, and to show how existing development artifacts can be used by an integrated documentation process.

0-7803-6431-7/00/$10.00 © 2000 IEEE

The Workflows

The Rational Unified Process defines the following *core workflows*, which categorize most of the work required to develop a software product:

- Business modeling
- Requirements
- Analysis and design
- Implementation
- Test
- Deployment

In addition to these workflows, which involve varying degrees of activity at different stages in the process, there are two more that are constant throughout the software process:

- Configuration and change management
- Project management

As they are currently described in the RUP, none of these workflows include the documentation aspects of software development: that is, they cover how to develop the software, but not how to develop documentation for the software's users.

As described in [3], the activities of a documentation team parallel the activities of a software team fairly closely. Essentially, documentation development has the same workflows as software development: designing, writing, testing, packaging, and managing.

While submitting documentation development to the same rigorous process as software development certainly buys a degree of control and repeatability, that is not enough: documentation development is not a standalone process, but one that depends on the software development process. A combined process needs to identify not only documentation's parallel workflows, but also how the documentation and software workflows integrate. The combined process needs to capture goals such as that the documentation accurately describes the software, and that the software does what the documentation describes.

The purpose of the user information, or documentation, workflow is to:

- Translate the use cases for the software into designs for online and print information, including organization and navigation.
- Implement the documentation design as a complete, consistent, and accurate information set.
- Verify that the information meets users' needs for using the product.
- Produce or deploy the information in a timely and efficient way.

In large, complex products or integrated sets of products, the documentation issues around scalability and extensibility can be quite complex. Shops that produce such information sets require style guides, templates and tools, and coordination across multiple groups, possibly in different locations.

0-7803-6431-7/00/$10.00 © 2000 IEEE

INTEGRATING THE SOFTWARE AND DOCUMENTATION WORKFLOWS

Since documentation development has the same types of workflow as software development, there is a strong mapping between the activities in the existing workflows and the required activities in a new documentation workflow.

Integration of software and documentation development can be addressed either by extending the existing software workflows to include documentation workers and activities, or by creating a new documentation workflow that is tightly integrated with multiple software workflows.

In the following sections, selected workers, activities, and artifacts of each software workflow will be examined and extended to cover documentation work. This should illustrate how the existing software workflows could be extended to include documentation activities.

After the treatment of the software workflows, the documentation workers and activities will be collected and summarized in terms of a single *documentation workflow*. Either view should be correct – your approach will depend on the degree of integration between your software group and writing group.

Software Workflows

Each of the software workflows will be discussed in terms of roles and artifacts that can be mapped to the needs of a documentation workflow.

The business modeling workflow

This is the first workflow to begin work in a software development process: it defines the main workers *business process analyst* and *business designer*.

The following artifacts, produced by the business process analyst and business designer, are of particular interest from a documentation perspective:

Business Process Analyst:

- Glossary, defining the language of the business domain.
- Business use case model, defining the general task flow of the processes being addressed by the software.

Business Designer:

- Business workers, business entities, and organization units, describing the users of the software and their organization.
- Business use cases, detailing the specific steps involving users and other entities in significant use cases.
- Business use case realizations, applying the steps to a specific set of users and entities.

0-7803-6431-7/00/$10.00 © 2000 IEEE

These artifacts can be mapped to an equivalent set of documentation artifacts, produced by a new role, the *information architect*:

Information Architect:

- Analysis artifacts:
  - Glossary, defining the terminology to be used in the documentation (information architect becomes another author of the existing glossary).
  - Task flow analysis, describing the tasks to be accomplished by the target audience, and the relationships among tasks
- Design artifacts:
  - Audience analysis, defining the target audience and its categories.
  - Preliminary task skeletons (essentially a plain-text form of a use case), detailing the specific steps involved at important stages of the task flow.
  - Preliminary tutorials, positioning the steps relative to a typical usage situation.

One of the most important mappings here is between use cases, and their equivalents in documentation: task flows. From a documentation point of view, use cases are one of the most important artifacts in the RUP. Use cases describe the possible flows of actions involved in accomplishing a particular task. This information, when consolidated from all use cases for a product, gives us the product's task flow (relationships among tasks), and at a more granular level, actual task documentation (steps needed to complete a task). Use-case instances, or scenarios, describe the possible sequences of actions for a particular use case. In documentation terms, these represent examples of how a task might actually be accomplished: in other words, a task scenario or tutorial.

While the mapping shown here is between use cases and task flow, some groups have successfully extended use case models to the point where full-fledged task documentation could be generated automatically from them, as described in [4].

Summary

Business Process Analyst	Business Process Designer	Information Architect
Glossary Business use case model		Glossary Task flow analysis
	Business workers and other entities Business use cases Business use case realizations	Audience analysis Task flows/task skeletons Tutorial skeletons

The Requirements Workflow

In the requirements workflow, the RUP defines the main workers *system analyst, architect,* and *user interface designer.*

The following artifacts, produced by the system analyst and user interface designer, are of particular interest from a documentation perspective:

System Analyst:

- Glossary (already described – the system analyst is another author of this document)

0-7803-6431-7/00/$10.00 © 2000 IEEE

- Business use-case model (already described – the system analyst is another author of this document)
- Use-case model, created from the business use-case model, summarizing the intended functionality of the proposed software.
- Requirements attributes, tracking specific user requirements and the use cases that implement them.
- Change requests, tracking change requests from the customer.

User Interface Designer:

- Actors, defining users of the system and which use cases apply to them.
- User-interface prototype, defining the look and feel of the user interface
- Use-case storyboard, defining the way in which use cases will be implemented by the user interface.

Some of these artifacts have already been defined in the previous workflow, and the result from a documentation perspective is merely a requirement to track additional inputs to the documentation plan.

Information Architect:

- Analysis artifacts:
  - Glossary (already described)
  - Task-flow analysis (already described)
  - Documentation requirements (from the software requirements – each software requirement that affects the user interface or task flows has a matching documentation requirement)
  - Change requests (from the software change requests – any change that affects the user interface or task flows has a matching documentation change request)
- UI designer artifacts:
  - Audience analysis (already described), expanded to describe which parts of the task flow apply to which audiences.
  - Documentation prototype, showing how the online information and any printed information will complement the look and feel of the software and be integrated with the user interface.
  - Documentation storyboard, showing the way in which the information will be used along with the software.
  - Sample tasks and tutorials, with UI details filled in.

0-7803-6431-7/00/$10.00 © 2000 IEEE

Summary (new artifacts only)

System Analyst	UI Designer	Information Architect
Use-case model Requirements Change requests		Updated task-flow analysis Documentation requirements
	Actors UI prototype Use-case storyboard	Audience-to-task mapping Documentation prototype Documentation storyboard Updated sample tasks and tutorials

The Analysis and Design Workflow
In the analysis and design workflow, the RUP defines the main workers *architect, designer,* and *database designer.*

The following artifacts, produced by the architect and designer, are of particular interest from a documentation perspective:

Architect:
- Analysis model, describing the organization of classes that can be inferred from the use case analysis
- Design model, doing the same thing but from a design perspective

Designer:
- Analysis and design classes (class names, relationships among classes, class methods, class data).
- Use-case realizations, describing how the classes interact to implement a particular use case.

Database Designer:
- Data model, describing the details of what data needs to be saved, or made persistent, by the software.

In this stage, the architect and designers develop a class and data model for the software. In many software processes, this information is considered completely irrelevant from a documentation perspective, since design elements such as the class hierarchy are not directly exposed in the user interface. However, the process that gives rise to the analysis and design classes and to the data model is actually quite similar to the process by which an information architect identifies key concepts requiring documentation, and requirements for reference information. While not all classes map to concepts requiring documentation, a significant portion of them are derived directly from the combination of use-case model and glossary, both produced in the business modeling phase. These are the same sources (task analysis and glossary) an information architect could use to identify key concepts.

0-7803-6431-7/00/$10.00 © 2000 IEEE

Information Architect:

- Concept model and concepts (from the class hierarchy and the glossary)
- Mapping of concepts to tasks (from the use-case realizations)
- Mapping of concepts to reference information (from class data, or the data model)
- Mapping of task steps to class methods (from class methods and use-case realizations)

While the mapping between classes based on key abstractions and their documentation equivalents is not exact, it provides an additional check on the equivalence of the software and documentation. By keeping track of the key abstraction classes in the software, a writer can make sure that changes in the understanding or treatment of these concepts are captured in the documentation.

To put it another way, both documentation concepts and software key abstraction classes are derived from the same sources: the glossary and use cases. By pairing the outputs from those sources (identifying the overlap between classes and concepts), we gain a mechanism for tracking design shifts that occur later in the software development process. When the interface to a key abstraction class changes, it signals a change in the way the software deals with the matching concept, and the concept's documentation can be re-examined, and updated if necessary.

The analysis and design models provide not only a treatment of concepts, but also mappings of concepts to reference information and concepts to tasks. The existence of a key abstraction class provides a concrete reference point for the software's treatment of some concepts; but the class's behavior in use cases and its handling of data provides relationships among concepts, tasks, and reference information.

Summary

Architect	Designers	Information Architect
Analysis and design model		Relationships among concepts
	Classes (with methods and data) Use-case realizations Data model	Mapping of concepts to tasks and reference information Relationships among reference information

The Information Architecture Emerges

We are at the point in the process where an information architecture has emerged. An *information architecture* is the organization and structure of information to meet the needs of users, designed with concern for aesthetic, functional, navigational, and retrieval issues.

Every documentation set has an information architecture, either explicit or implicit. As described in [3], a well-defined, explicit information architecture should provide the following benefits:

- It organizes information according to the needs of the users.
- It enables the production of an aesthetically pleasing presentation of information.
- By providing a consistent organizational scheme, it helps users understand the product and more quickly internalize a model of its functionality.

0-7803-6431-7/00/$10.00 © 2000 IEEE

- By providing a consistent set of navigable relationships, it allows users to find their way easily to the information they are seeking and to move quickly to related information.
- By building up from basic types and relationships into more complex categories and inter-relationships, the architecture enables the information to be scaled up into larger sets, and to be decomposed into components that can be reused in different contexts or for different deployment strategies.

For example, one Rational group articulated a role-based, task-oriented documentation architecture, structuring the information by types of user (actor) and by the tasks and scenarios that represent those users' work. By developing templates for task information, the group could reconfigure information for printed manuals or online help as necessary. Use case elaborations could be used directly in developing the task modules, and possibilities opened up for sharing information modules across groups with similar information organizations, or with course developers, or with usability engineers for test scenarios.

In the process described in this paper, the information architect produces an information-typed architecture that categorizes information as tasks, concepts, or reference material.

Information Architecture	Software Architecture
Glossary	
Task Flow	Use case model
Tasks	Use cases
Task steps	Class methods
Tutorials	Use-case realizations
Concept Hierarchy	Class hierarchy
Concepts	Classes
Reference Model	Data model
Reference information	Class data

In the preceding list, the task flow, concept hierarchy, and reference model form an *information architecture* for the documentation. They identify the pieces of documentation needed to support a customer using the software in the same way that the *software architecture* defines the pieces of code needed to create the software in the first place.

This information typing model is particularly suited to a structured process for object-oriented software development, because software development requires analysis of the problem domain in exactly the same terms: concepts (classes), tasks (methods), and reference (data), with relationships among them captured by models (analysis, design, data, and use case). The mapping between the two treatments is unlikely to be perfect: just as not all classes represent concepts, so not all methods and data represent tasks or reference information; and from the other side, there will always be documentation that has no exact equivalent in the system (such as installation tasks). However, by identifying the overlap, we can track conformance between software and documentation and eliminate redundant effort by coordinating the work of the information architect and the software architect.

0-7803-6431-7/00/$10.00 © 2000 IEEE

THE IMPLEMENTATION WORKFLOW, THE TEST WORKFLOW, AND THE DEPLOYMENT WORKFLOW

The next three workflows, while certainly important, do not integrate the software and documentation processes as much as the first three.

The following additional workers are needed for these workflows:

Implementation Workflow

- The *writer*, who implements the information architecture
- The *technical illustrator*, who provides illustrations
- The *editor*, who provides quality control

Test Workflow

- The *documentation tester*, who tests the functionality and integration of the information

Deployment Workflow

- The *production specialist*, who administrates any automated documentation build processes
- The *page designer*, who develops templates and enforces look-and-feel consistency in printed and online documentation sets

THE CONFIGURATION AND CHANGE MANAGEMENT WORKFLOW, AND THE PROJECT MANAGEMENT WORKFLOW

The last two workflows, in effect throughout the development process, require the following documentation-specific roles:

- The *documentation tools developer*, who develops any project-specific tools and processes to support the documentation plan
- The *documentation manager*, who develops a documentation project plan and coordinates schedules and resources with the other manager roles

THE DOCUMENTATION WORKFLOW

The preceding workflow-based treatment concentrated on those documentation artifacts that had a software process equivalent, and on describing the integration points between the two processes. The following phase-based treatment describes each of the roles in turn, as part of a *documentation workflow*, and then summarizes the activities for each role based on the development *phase*.

The RUP defines a phase as "The time between two major project milestones, during which a well-defined set of objectives is met, artifacts are completed, and decisions are made to move or not move into the next phase." [2, glossary].

0-7803-6431-7/00/$10.00 © 2000 IEEE

The RUP defines the following phases for a software development process:

- Inception (gathering requirements, building a vision)
- Elaboration (designing the software)
- Construction (building the software)
- Transition (shipping the software)

Worker activities can vary considerably from phase to phase: for example, a build manager has little activity during the inception phase, while an architect has (hopefully) little activity during the transition phase (though they may have already started on the inception phase of the next cycle).

The list of workers in a documentation group is long, but in fact most of these roles map to existing roles in the Rational Unified Process. In many groups individuals play multiple roles, and some roles are more or less important depending on the size of the group and the nature of the software.

Information Architect

The information architect is responsible for developing and maintaining the *information architecture document*, specifying the overall organizational, structural, and navigational properties of the information set. Like the software architect, the information architect's view is broad rather than deep. The information architect oversees the maintenance of the architectural integrity of the information set through implementation and deployment, and through successive versions of the software. The information architect is a reviewer of the software architecture document, and the software architect is a reviewer of the information architecture document.

The information architect determines the components of online and printed documentation that will be implemented by the documentation team. The information architect develops and maintains the *documentation project plan*, specifying what documents will be produced in the current development cycle.

Technical Writer

The technical writer is responsible for developing the *documentation plan*, or specification, for each information component he or she produces. The writer writes the document and ensures that it is reviewed and/or tested according to the plan. This role is an *implementer* role.

Technical Editor

Also an *implementer* role, the technical editor is responsible for the style and consistency of documentation across the entire information set. The editor is also responsible for the *editorial style guide* that defines rules and guidelines for style across the information set.

0-7803-6431-7/00/$10.00 © 2000 IEEE

Document Reviewer/Tester

Similar to quality assurance for the bits, someone must validate that the documentation does the job it was designed to do. Document reviews are commonly part of the software development process. In some shops, the quality engineers may test the help and automated tutorials. In some cases, usability testing may include documentation. Thus, the artifacts may include review comments, QA plans and reports, and usability plans and reports, and change requests may be filed against the documentation. This is a *tester* role.

Documentation Manager

Like the project manager, the documentation manager allocates documentation resources; shapes documentation priorities; and coordinates with the project manager on the goals, activities, and schedule for the documentation deliverables. This information is part of the *documentation project plan*. The documentation manager also establishes the set of practices and tools to ensure the integrity and quality of the documentation artifacts.

Tool Supporter

Large documentation groups with a complex set of online and print deliverables often require programming support to integrate the various documentation tools used within the group and to produce the information artifacts in the formats required by the product. Tools requirements and plans are defined in a *tool support assessment* document. This is an *implementer* role.

Production Specialist

Production specialists, an *implementer* role, build the source files in the deliverable documentation artifacts (printed books, help systems), check the completeness and quality of the artifacts, and deliver the files to print vendors or to the software builds, as specified in the documentation project plan. The production specialists are responsible for the *Bill of Materials*, or *BOM*, for the information set.

Page Designer

The page (or screen, or book) designer is responsible for consistent look and feel of the user information set, including printed documents and online information. The page designer produces a book design, and is responsible for templates and guidelines for producing information according to the design. This is an *implementer* role.

Technical Illustrator

The illustrator is responsible for producing clear and consistent graphics for the user information set, including printed documents and online information. The illustrator develops guidelines and rules to ensure a consistent set of illustrations across the information set. This is an *implementer* role.

0-7803-6431-7/00/$10.00 © 2000 IEEE

DOCUMENTATION PHASES

Inception Phase

In this phase of a project, the documentation workers identify the scope of the documentation work for the project, review and contribute to various planning artifacts, and gain the concurrence of stakeholders for documentation planning. The following table identifies the documentation activities and the artifacts that are affected by documentation planning. A new artifact, the documentation project plan, is produced. The doc workers are added to the review lists for existing artifacts and may contribute sections to those artifacts.

Activities	Worker	Artifacts (outputs)
Analyze audience to identify users	Information architect	Vision document
Analyze competitors' information sets and state of the art in documentation tools/techniques	Documentation project manager, Information architect	Business case
Identify changes to the information set required for new features	Documentation project manager	Requirements
Analyze existing product and documentation defects	Documentation project manager	Requirements
Identify delivery requirements (formats, media)	Documentation project manager, Information architect	Requirements
Develop coarse-grained documentation plan	Documentation project manager	Documentation project plan
Develop documentation test goals: reviews, usability, editing, functional	Documentation project manager	Documentation project plan
Develop documentation business case	Documentation project manager	Business plan
Review project artifacts	All	All
Start project glossary	Information architect, with software architect	Glossary
Identify impact of localization requirements	Documentation project manager, information architect	Requirements, business plan, documentation project plan
Identify requirements for doc tools needed to support online help, tutorials, production	Tool specialist	Tool support assessment, documentation project plan

0-7803-6431-7/00/$10.00 © 2000 IEEE

Elaboration Phase
In this phase of a project, documentation workers make detailed decisions about the information architecture and design work needed, and develop the plan for the information set. At the end of this phase, the scope of work and risks are known, and implementation can begin.

The new artifacts for the information workflow are:

- Information architecture specification, which parallels the software development plan
- The individual doc plans
- The book/page design specification

The other artifacts listed in the table already exist in the RUP but now contain additional sections, supplied by the documentation group, about user information requirements and considerations.

Activities	Worker	Input Artifacts	Output Artifacts
Develop information architecture, or review and modify existing architecture as needed.	Information architect	Vision document (use case model), additional user definition information as needed.	Information architecture specification
Determine what modules of information will be developed and their output formats	Information designer	Vision document (use case model), information architecture specification	Documentation project plan
Develop resource, staffing plan	Documentation project manager	Requirements, information architecture specification, tool support assessment	Documentation project plan
Plan for manufacturing	Documentation project manager	Business plan, requirements for delivery	Documentation project plan, draft BOM
Identify tools and methods plan	Documentation project manager, Tool specialist	Requirements for tools	Documentation project plan, tool support assessment
Review, update style guides and templates	Editor, production specialist	Information architecture specification, documentation project plan	Editorial style guide, doc templates

0-7803-6431-7/00/$10.00 © 2000 IEEE

Produce final doc plan	Documentation project manager, information designer		Documentation project plan
Develop doc test plan to ensure that info is complete, accurate, usable, accessible	Usability engineer, QA engineer	Vision document (use case model), documentation project plan, requirements	Test plans
Identify dependencies, including UI requirements for text, error messages, etc.	Documentation project manager	Risk management plan	Risk list
Elaborate doc-related requirements	Documentation project manager	Documentation project plan	Requirements
Produce detailed iteration plan and schedule	Documentation project manager	Software project plan	Documentation project plan
Develop plans for individual documents	Writer, editor	Information architecture specification, documentation project plan	Doc plans
Develop plans for doc tools support	Tool specialist	Documentation project plan, tool support assessment	Tools specification
Develop book/page design specification	Book designer, illustrator	Information architecture specification, documentation project plan	Book design specification
Update editorial style guide	Editor	Information architecture specification, documentation project plan, individual doc plans	Editorial style guide
Develop the build plan	Production specialist	Documentation project plan, individual doc plans	Build plan
Develop localization plan for documents	Documentation project manager	Information architecture specification, documentation project plan, individual doc plans	Localization plan

0-7803-6431-7/00/$10.00 © 2000 IEEE

Construction Phase
In this phase of a project, the documents are written and produced as needed for review, beta testing, etc. Help files are integrated with the software and tested as needed.

The new artifacts for the information workflow are:

- The manuals, help files, web content, etc. produced by the documentation group
- The art files

The other artifacts listed in the table already exist in the RUP but now contain additional sections, supplied by the documentation group, about user information requirements and considerations.

Activities	Worker	Artifacts
Write the information: manuals, help topics, components, modules, fix defects	Writer, editor	Manuals, help files, web content, etc.
Develop support tools	Tool specialist	Tools
Execute the build plan	Writer, editor, production specialist, tool specialist	Test results
Test the manufacturing process	Production specialist, tool specialist	Test results
Execute the doc testing plan	Reviewers, QA engineers, usability engineers, documentation testers, editors	Test results
Complete the Bill of Materials	Production specialist	BOM
Execute internal deployment plan	Production specialist	Test results
Integrate art package	Technical illustrator	Art files

Transition phase
In this phase of a project, the information set is deployed (with the rest of the software product) to the users.

There are no new artifacts for the information workflow in this phase. The artifacts listed in the table already exist in the RUP but now contain additions, supplied by the documentation group, about user information requirements and considerations.

Activities	Worker	Artifacts
Execute the beta plan	All	Product
Release to manufacturing	Production specialist	Product
Produce the localization release	Production specialist	Localization kit

0-7803-6431-7/00/$10.00 © 2000 IEEE

SUMMARY

Adding documentation to the Rational Unified Process is not as complex as the previous tables might make it appear. Documentation processes closely parallel the analyze-design-implement-test-release sequence of activities that programmers follow to write the code. However, the documentation workers require different skills than the software development workers, and the artifacts produced are different from the bits, however tightly integrated. In addition, information development has dependencies on the code development. So it is worthwhile to look at information development as a separate process, while at the same time preserving points of integration and making use of the common treatments of key concepts, task analyses, and data models.

Changes to Existing Workers, Artifacts, and Workflows

As noted in the section on documentation workers and artifacts, the information development roles must be elaborated beyond *technical writer*. As is always the case, these are roles, not job titles: multiple roles are likely to be performed by individuals. The artifacts may be similarly combined. But all documentation groups must address organization, structure, and navigation of information; production of help, online text, and manuals; consistent style and formatting; artwork; delivery media; and so forth. Making the roles, activities, and artifacts part of the RUP will ensure smoother and better information development, and a higher-quality product overall.

In particular, use cases can benefit information design and development by helping documentation groups focus on essential interactions with the system. Conversely, use case development may also benefit from the involvement of technical writers, who bring a user focus driven by their more direct accountability: bad documentation is immediately noticeable to a user, where a bad system design may take several iterations to become exposed.

This paper identifies integration points between a software development process and a documentation development process. In order for this integration to take place, the following first steps need to be in place:

- The software development process must be articulated and adopted
- A documentation development process must be articulated and adopted
- The documentation group's involvement in the software process must start at the beginning; many of the integration points identified occur in early phases of the process.
- The documentation and software development groups must share ownership of key documents, in particular the glossary and use cases.

While each of these steps has rewards in itself, the final pay-off is provided by the integration of the two processes, which eliminates redundant work, ensures consistency of terminology and functionality, and makes use of the strengths of both software and documentation worker in the shared development of their deliverables.

Next Steps

User information is one piece of the "user experience" or the "user interaction" aspects of software products. This workflow could be expanded to include UI design and usability evaluation, GUI screen text development and maintenance, and courseware. The existing RUP role of *user interface designer* is an excellent start at this, but does not fully explore the integration of user interface and documentation, which is the place where consistency between software and documentation is most important, and most harmful when missing.

0-7803-6431-7/00/$10.00 © 2000 IEEE

Traditionally, different groups are responsible for these activities and artifacts, but users' ability to make efficient and effective use of the software is negatively affected if these information sources are inconsistent, redundant, contradictory, or leave gaps for the users to fall through. If the problems are big enough, expensive support calls and lost business can result.

Expanding the Rational Unified Process to explicitly take the whole user experience into account would further improve the quality of software products.

ACKNOWLEDGEMENTS

We would like to acknowledge the help of the following reviewers and contributors:

- Karl Hakkarainen and Liz Augustine of Rational Software
- The RUP team at Rational Software

REFERENCES

[1] Kruchten, Philippe. The Rational Unified Process: an Introduction. 2nd edition. Addison-Wesley, 2000.

[2] Rational Unified Process. < http://www.rational.com/rup/index.jhtml >

[3] Utt, M.H., and R. Mathews. "Developing a User Information Architecture for Rational's ClearCase Product Family Documentation Set." Conference Proceedings; ACM SIGDOC 1999, pages 86-92.

[4] Cécile Paris, Nadine Ozkan, and Flor Bonifacio. "Novel Help for On-line Help." Conference Proceedings, ACM SIGDOC 1998. pp. 70 – 79.

ABOUT THE AUTHORS

Michael Priestly is an information developer for the IBM Toronto Software Development Laboratory. He has written numerous papers on subjects such as hypertext navigation, singlesourcing, and interfaces to dynamic documents. He is currently working on XML and XSL prototypes for help and documentation management.

Mary Hunter Utt is a web user interface engineer at SiteScape, Inc. She has considerable experience in UI design/usabililty, information architecture, and web application development.

0-7803-6431-7/00/$10.00 © 2000 IEEE